Trabajo final de trimestre

Tutoriales de Apache servidor, Vsftpd y Git

Índice

02 Apache Virtual Hosting

02 Apache Virtual Hosting
¿Que es?

Para poder hacer funcionar mas de una página web a la vez necesitan ser hosteadas en un servidor para poder tenerlas guardadas. Con el Virtual Hosting de Apache podemos conseguir hostear varias páginas en un mismo servidor con la misma ip. Además de esto es fácilmente configurable.

Esta funcionalidad es lo que hace que Apache Virtual Hosting sea una de las herramientas mas usadas a la hora de crear dominios y páginas web.

02 Apache Virtual Hosting
Pre-requisitos

Primero que nada, hay que instalar el servidor web de apache en nuestra consola Linux con el comando:

- **sudo apt-get install apache2**

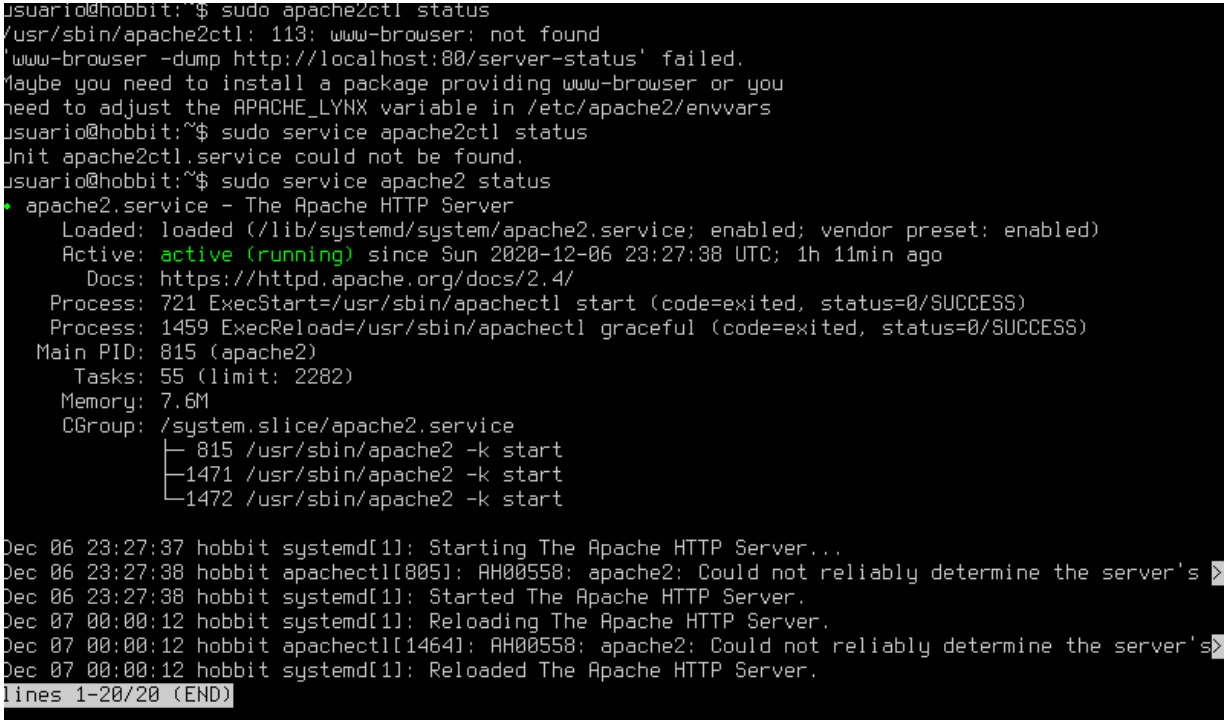
  Y para gestionar el servicio se usaría:

- **Apache2ctl [-k start|restart|graceful|graceful-stop|stop]**

  Siendo los principales start para empezar, restart para reiniciar y stop para parar el servicio. Graceful es un reinicio suave, se terminan de servir las peticiones que están establecidas y cuando se finaliza se hace un reinicio.

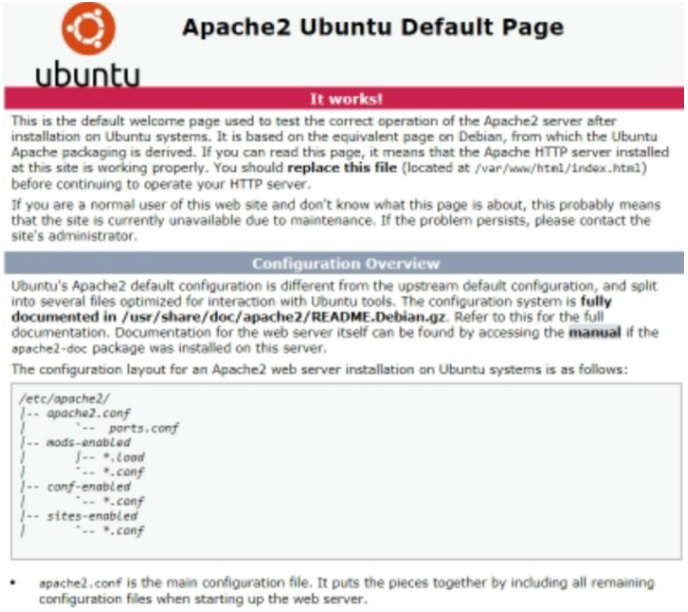
02 Apache Virtual Hosting

Cuando miremos el server status nos debería de salir algo como:



02 Apache Virtual Hosting

Y al ejecutar la ip del servidor nos encontramos la página de ejemplo.



Apache2 Ubuntu Default Page

ubuntu

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.

02 Apache Virtual Hosting

Si queremos configurar o cambiar el contenido de la página a uno personalizado, lo que tenemos que hacer estos comandos.

- sudo mkdir /var/sudo mkdir /var/www/example.com para crear una carpeta nueva
- sudo cd /var/www/example.com para meternos en la carpeta
- sudo nano index.html y creamos un html normal como queramos
- Luego, nos vamos a etc/apache2/sites-availables/ de nuevo y hemos copiado el ejemplo de default, le ponemos nuestro nombre y lo modificamos, modificando server-name (nombre del server), server-alias (el alias de nuestra página) y el server-document-root (donde esta nuestra carpeta con nuestros archivos)

02 Apache Virtual Hosting

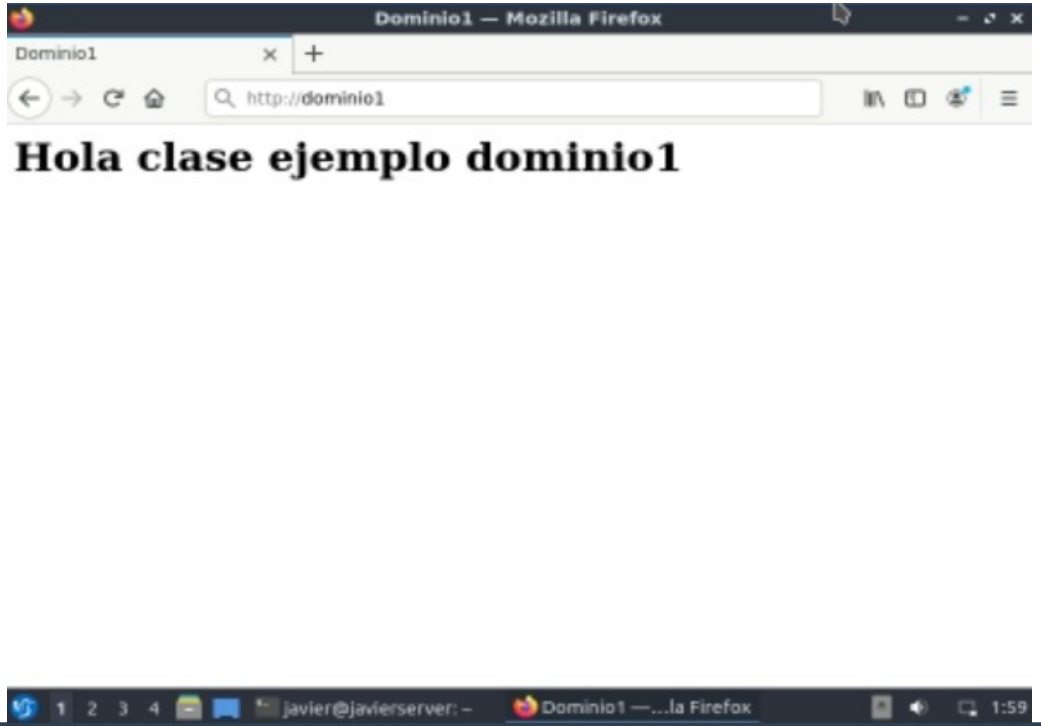
Y luego escribimos sudo a2ensite *nuestro dominio*.config para poder habilitarlo y usamos el sudo a2dissite 000-default.config para deshabilitar la página de ejemplo.

Hacemos un sudo apache2ctl restart para reiniciar apache.

Y para finalizar vamos a nuestra página siendo esta http://*nombre de página* y nos debería de entrar en el html creado

02 Apache Virtual Hosting

El resultado saldría así



03 Apache Mapeo de URL

03 Apache mapeo de URL

¿Que es?

El mapeo de URL es una herramienta que se utiliza para redireccionar las peticiones entrantes a una URL diferente. Este redireccionamiento se produce antes que cualquier otro procesamiento de peticiones entrantes.

03 Apache mapeo de URL

Primero que nada nos metemos en la carpeta sites-availabre, dentro de apache2, para luego poner nano apache1.conf y si bajamos nos encontraremos esto

03 Apache mapeo de URL

¿Esto nos resulta familiar, verdad? Bueno, en el siguiente paso toca escribir un poco. Debajo de DocumentRoot

<Directory /var/www/apache1>

Options -Indexes

</Directory>

Lo que estamos haciendo con esto es sobreescribir las opciones para todos los directorios y subdirectorios de var/www/apache1.

03 Apache mapeo de URL

Para que cambiamos esto os estaréis preguntando. Cuando no tenemos un archivo index.html, lo que hace el sistema es indexar automáticamente, es decir que nos aparece todos los ficheros que disponemos. Lo que hemos cambiado es para que, si no hay un archivo index.html, no nos muestro todos nuestros archivos, ya que eso sería un fallo de seguridad bastante grande, si no que en su lugar nos muestra forbidden.

"No tienes permiso para acceder/estar en este servidor"

03 Apache mapeo de URL

Ahora vamos a encargarnos de los enlaces simbólicos que es un enlace que nos redirige a un fichero que se encuentra en un lugar distinto de nuestra estructura de directorios. Queremos que nuestra página se adhiera a los archivos que nosotros queremos, así que habrá que poner el comando Options -FollowSymLinks. Ahora cuando busquemos el enlace simbólico nos dará otra vez forbidden.

03 Apache mapeo de URL
Alias

El alias nos permite definir parte de la url que cuando se acceda a ella, los recursos se obtienen no de DocumentRoot si no de otra parte. Un ejemplo sería que las fotos las queremos coger de una carpeta fotos, pues se pone Alias *ruta del directorio* debajo de DocumentRoot.

```
DocumentRoot /var/www/apache1
 Alias /web /home/debian/directorio
```

“Ejemplo de alias”

03 Apache mapeo de URL
Negociación de contenido

Alguna vez cuando estamos navegando en internet se nos presenta algo como esto 

. Si sabemos que estamos en Google, pero que significa lo de “es”. Eso es lo que vamos a ver en esta parte.

La negociación de contenido nos permite ofrecer diferentes versiones del mismo archivo dependiendo de algunas condiciones como por ejemplo, el idioma.

Lo aremos mediante un archivo .var

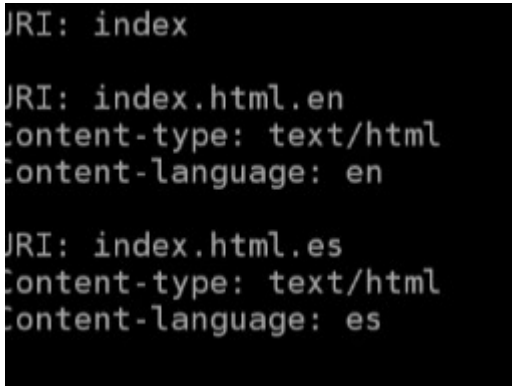
03 Apache mapeo de URL

Primero crearemos una carpeta dentro de var/www/apache1 conel nombre que queramos. Y ahí crearemos 3 archivos:

- Index.html.es
- Index.html.en
- Index.html.var

03 Apache mapeo de URL

Añadimos texto en los dos primeros html. En el primero al ser es pondremos texto en español y en el segundo al ser en lo pondremos en ingles. Dentro del .var pondremos esto



- URI: index significa que se usara como Index
- Content-type: text/html para saber que tipo de archivo es.
- Content-language: para saber en que lenguaje está para que el navegador lo seleccione.

03 Apache mapeo de URL

Luego dentro del apache1 usaremos una clausula

<Directory /var/www/*nombre de tu sitio*>

```
    DirectoryIndex index.var
    AddHandler type-map .var
```

<Directory>

DirectoryIndex index.var se usa para que el archivo .var sea usado como index y AddHandler es un manejador, que es nuestra forma de decirle a Apache como tratar los archivos, en este caso mavos a decir que al archivo .var lo vamos a tratar como un mapa.

03 Apache mapeo de URL

Ahora dependiendo del idioma del navegador nos pondrá una página u otra. Como hemos puesto en ingles y en español variará según esos dos idiomas.

03 Apache mapeo de URL
Redirecciones

Esta opción se utiliza para cuando, por ejemplo, un cliente busca un recurso (foto,video…) y ya no se encuentra en esa ubicación, así que el cliente tendrá que pedir el recurso para la nueva ubicación.

Hay dos tipos de redirecciones:

- Temporales: el recurso se ha movido pero volverá a su sitio, así que le redireccionamos a donde esté el archivo.
- Permanentes: el recurso se ha movido para siempre así que ya no estará en la carpeta origen

03 Apache mapeo de URL

Para usar las redirecciones simplemente usaremos el comando Redirect que se pone debajo del DocumentRoot.

Su estructura es la siguiente:

Redirect “/Web1”(lo que busca el usuario) “Destino”(a donde se le redirige.

Redirect actua de forma que si el usuario intenta entrar en web1 le va a llevar al destino que nosotros pongamos a la derecha.

Si queremos que esa redirección sea permanente simplemente ponemos antes de web1 la palabra permanent o 301 el (codigo de estado).

03 Apache mapeo de URL

Por ejemplo aquí estamos diciendo que cuando queramos acceder a buscar nos lleve directamente a la página de Google.

```
ServerAdmin webmaster@localhost
DocumentRoot /var/www/html
Redirect "/buscar" "https://www.google.es"
```

03 Apache mapeo de URL
Páginas de errores personalizadas

Pongamos el ejemplo de que nos salte un error 404 not found. Esa página queda bastante sosa. Pues en esta sección vamos a ver como, dependiendo del error que nos de el servidor mostremos una página u otra.

03 Apache mapeo de URL

Ahora nos meteremos otra vez en etc, pero en vez de en

sites-available nos meteremos en conf-available y nos meteremos en localiced-error.pages.

```
javier@javier-VirtualBox:/etc/apache2/sites-available$ cd ..
javier@javier-VirtualBox:/etc/apache2$ cd conf-
conf-available/ conf-enabled/
javier@javier-VirtualBox:/etc/apache2$ cd conf-available/
javier@javier-VirtualBox:/etc/apache2/conf-available$ ls
charset.conf  localized-error-pages.conf  other-vhosts-access-log.conf  security.conf  serve-cgi-bin.conf
```

03 Apache mapeo de URL

Cuando nos metamos nos saldrán varias líneas de las cuales solo nos interesan:

- Errores de ejemplo que afectan a todo el virtual host.

```
#
# Some examples:
#ErrorDocument 500 "The server made a boo boo."
#ErrorDocument 404 /missing.html
#ErrorDocument 404 "/cgi-bin/missing_handler.pl"
#ErrorDocument 402 http://www.example.com/subscription_info.html
#
```

03 Apache mapeo de URL

- Estas lineas lo que contienen son un alias y un redireccionamiento para que cada error sea traducido según el idioma

```
#<IfModule mod_negotiation.c>
#       <IfModule mod_include.c>
#               <IfModule mod_alias.c>
#
#                       Alias /error/ "/usr/share/apache2/error/"
#
#                       <Directory "/usr/share/apache2/error">
#                               Options IncludesNoExec
#                               AddOutputFilter Includes html
#                               AddHandler type-map var
#                               Order allow,deny
#                               Allow from all
#                               LanguagePriority en cs de es fr it nl sv pt-br ro
#                               ForceLanguagePriority Prefer Fallback
#                       </Directory>
#
#                       ErrorDocument 400 /error/HTTP_BAD_REQUEST.html.var
#                       ErrorDocument 401 /error/HTTP_UNAUTHORIZED.html.var
#                       ErrorDocument 403 /error/HTTP_FORBIDDEN.html.var
#                       ErrorDocument 404 /error/HTTP_NOT_FOUND.html.var
#                       ErrorDocument 405 /error/HTTP_METHOD_NOT_ALLOWED.html.var
#                       ErrorDocument 408 /error/HTTP_REQUEST_TIME_OUT.html.var
#                       ErrorDocument 410 /error/HTTP_GONE.html.var
#                       ErrorDocument 411 /error/HTTP_LENGTH_REQUIRED.html.var
#                       ErrorDocument 412 /error/HTTP_PRECONDITION_FAILED.html.var
#                       ErrorDocument 413 /error/HTTP_REQUEST_ENTITY_TOO_LARGE.html.var
#                       ErrorDocument 414 /error/HTTP_REQUEST_URI_TOO_LARGE.html.var
#                       ErrorDocument 415 /error/HTTP_UNSUPPORTED_MEDIA_TYPE.html.var
#                       ErrorDocument 500 /error/HTTP_INTERNAL_SERVER_ERROR.html.var
#                       ErrorDocument 501 /error/HTTP_NOT_IMPLEMENTED.html.var
#                       ErrorDocument 502 /error/HTTP_BAD_GATEWAY.html.var
#                       ErrorDocument 503 /error/HTTP_SERVICE_UNAVAILABLE.html.var
#                       ErrorDocument 506 /error/HTTP_VARIANT_ALSO_VARIES.html.var
#               </IfModule>
#       </IfModule>
#</IfModule>
```

03 Apache mapeo de URL

Descomentamos todas esas líneas y nos fijaremos en ese alias. Ese alias nos permite ofrecer las páginas en un directorio. Como dijimos antes en segundo lugar se pondrá la carpeta de errores para que nos redirija allí . Y por último debemos de incorporar el módulo de include con el comando a2enmod include.

04 Apache Control de acceso y auntenticación

04 Apache Control de acceso y auntenticación

¿Que es?

El control de acceso se refiere a lo que podemos hacer para que el servidor distinga quien y a donde puede entrar discriminando por ejemplo entre tipos de usuarios, IP y en nombres de dominios.

Para el control de acceso usaremos el atributo Require.

Los valores que puede tomar Require son:

- Require all granted: El acceso es permitido siempre da igual quien sea.
- Require all denied: El acceso es denegado siempre.
- Require user userid [userid] ...: El acceso es permitido sólo si los usuarios indicados se han registrado.
- Require group group-name [group-name] ...: El acceso es permitido sólo a unos grupos de usuarios específicos.
- Require valid-user: El acceso es permitido a los usuarios válidos.
- Require ip *Cualquier ip*: El acceso es permitido si se hace desde el conjunto de direcciones especificadas.
- Require host dominio: El acceso es permitido si se hace desde el dominio adecuado.
- Require local: El acceso es permitido solo desde localhost.

04 Apache Control de acceso y auntenticación

- Hay que denotar tambien que se puede poner not delante del require para negar esa linea, por ejemplo Require not local hará que no podamos acceder desde el localhost

  También hay políticas de acceso, que son:

- RequireAll: requiere que se cumplan todas.
- RequireAny: requiere que solo se cumpla 1.
- RequireNone: no requiere que se cumpla ninguna de las condiciones que hayamos puesto

04 Apache Control de acceso y auntenticación

Control de acceso

Como ejemplo, vamos a crear dos carpetas dentro var/www/ con un nombre cualquiera cada una.

Ahora nos metemos en sites-available y nos metemos en nuestro .conf

Ahora pondremos una clausula directory con la ruta del archivo al que le queramos poner el require.

04 Apache Control de acceso y auntenticación

¿Os acordáis de las carpetas que creamos? Pues vamos a introducirles unos index.html que sean distintos para reconocerlos.

Luego volvemos al .conf y yo, por ejemplo le he puesto esto:

```
<Directory /var/www/*nombre carpeta1*/>
        Order Allow,Deny
        Allow from *ip*
</Directory>


<Directory /var/www/*nombre carpeta2*/>
        Require local
        RequireAll
</Directory>
```

04 Apache Control de acceso y auntenticación

- La primera orden dice que requiere una ip determinada para entrar en la carpeta 1
- La segunda dice que queremos se llegue desde localhost y que se necesita tener todas los requerimientos para poder entrar

04 Apache Control de acceso y auntenticación

Autenticación básica

Vamos ahora con la autenticación básica. Esta se trata de que vamos a tener nuestros usuarios en un fichero de texto plano con sus contraseñas (Encriptadas con hash). Tiene un problema, y es que nosotros guardamos todo como texto plano y es poco seguro así que se usa poco.

Primero vamos a crear una carpeta en var/www/*nombre dominio*/ donde guardaremos un index.html para el cual necesitaremos una auntenticacion.

04 Apache Control de acceso y auntenticación

Para esto usaremos otra vez Directory con los atributos:

<Directory var/www/*nombre dominio*/>

- AuthUserFile “etc/apache2/*carpeta donde guardemos claves/passwd.txt” para acceder al txt donde están los usuarios con sus contraseñas.
- AuthName “Ejemplo” para que solo pueda acceder el usuario con ese nombre
- AuthType Basic porque estamos con autenticación básica

04 Apache Control de acceso y auntenticación

- Tambien podemos ponerlo para grupos usando
- AuthGroupFile para seleccionar los grupos de usuarios
- Require group para decir que grupo requiere.

  Para crear los archivos de usuarios se usa la instrucción

  “htpasswd[-c] etc/apache2/*carpeta donde guardemos claves/passwd.txt usuario1” y posteriormente nos pedirá una contraseña para asociarla a ese usuario.
- Para crear grupos se usa la misma instrucción pero al final se pone nombregrupo: usuario1 usuario2 usuario3

04 Apache Control de acceso y auntenticación

Autenticación digest

La autenticación digest es un tipo de autenticación que ha desbancado a la autenticación básica ya que se cifra la contraseña que se envía. Además de esto, dependiendo de la hora a la que se le mande la contraseña al servidor va a estar cifrada de una manera diferente haciéndolo muy segura. Para activarla primero debemos de poner el comando a2enmod auth_digest.

04 Apache Control de acceso y auntenticación

Se usa igual que la del tipo basic pero con algunos cambios:

<Directory var/www/*nombre dominio*/>

- AuthUserFile “etc/apache2/*carpeta donde guardemos claves/passwd.txt” igual que en la basic
- AuthName “Nombre del dominio” el tipo digest usa tu dominio para el cifrado, lo explicaré a la hora de crear usuarios
- AuthType Digest para que el tipo sea digest

El resto es igual.

04 Apache Control de acceso y auntenticación

A la hora de crear un usuario nuevo es donde entra en juego lo que dije en la anterior página. Ya que cada usuario se guarda en un dominio, debemos especificarlo a la hora de crear el usuario.

Con el comando htdigest -c /etc/claves/*nombre fichero*.txt *nombre dominio* usuario1.

04 Apache Control de acceso y auntenticación

Políticas y control de acceso

Pongamos que hemos usado varios tipos de requerimientos para un archivo.¿Como se va a ejecutar eso?¿Que requerimientotiene mas prioridad?Pues estás políticas pueden indicarse de una forma muy parecida a los Require ya vistos:

- RequireAll: requiere que se cumplan todas.
- RequireAny: requiere que solo se cumpla 1.
- RequireNone: no requiere que se cumpla ninguna de las condiciones del bloque de condiciones

04 Apache Control de acceso y auntenticación

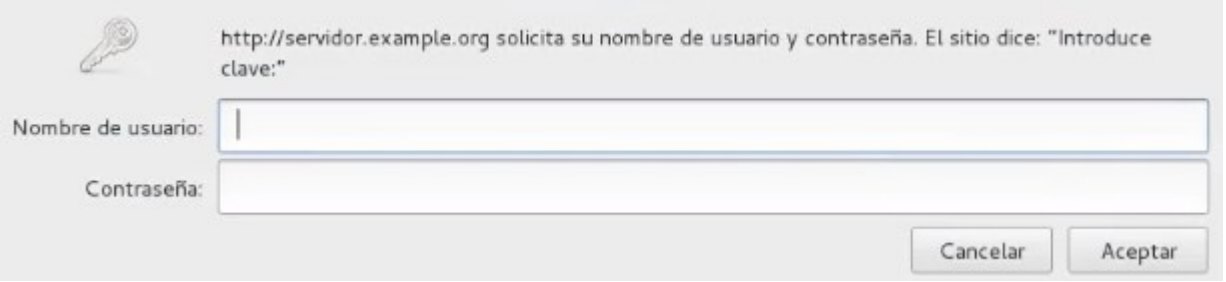
Lo bueno de las políticas de acceso es que su uso es muy sencillo.

- Para poder entrar en el directorio deseado se necesitara que se cumpla el primer bloque de condiciones entero y que no se cumpla ninguno del segundo

```
<Directory *Ruta del directorio*>
        <RequireAll>
                Require ip *ip*
                Require local
        </RequireAll>
        <RequireNone>
                Require ip *otra ip*
                Require valid-user
        </RequireNone>
</Directory>
```

04 Apache Control de acceso y auntenticación

Uno de los detalles de las políticas es que se leen secuencialmente, es decir, se leen de arriba hacia abajo y si se cumple la primera condicion, deja de leer y te da el acceso y si no pasa a la siguiente y así hasta que acabe. Si nos pide que nos identifiquemos nos pondrá un cuadro tal que así.



VSFTPD o Very Secure FTP DAEMON

VSFTPD o Very Secure FTP DAEMON

¿Que es?

VSFTPD es un servidor FTP para Linux y demás UNIX. Como el nombre indica, se trata de un servidor de este tipo mucho más seguro que los estándares, además de que ofrece varias opciones interesantes.

Como toda aplicación, previamente deberemos instalarla usando el comando:

- sudo apt install vsftpd

VSFTPD o Very Secure FTP DAEMON

Como configurarlo

Ahora toca configurarlo, ponemos el comando:

- sudo nano /etc/vsftpd.conf

Y nos saldrá algo como:

VSFTPD o Very Secure FTP DAEMON

Ahora vamos a ir descomentando (Borrar las almohadillas) poco a poco los comandos:

- write_enable: YES

```
# Uncomment this to enable any form of FTP write command.
#write_enable=YES
```

- chroot_local_user=YES

```
# You may restrict local users to their home directories.  See the FAQ
# the possible risks in this before using chroot_local_user or
# chroot_list_enable below.
#chroot_local_user=YES
```

Ahora deberemos escribir lo siguiente para que la configuración funcione con el usuario actual y con cualquier otro.

- user_sub_token=$USER
- local_root=/home/$USER/ftp

VSFTPD o Very Secure FTP DAEMON

Para que pueda haber muchas conexiones limitaremos la cantidad de puertos utilizados en el archivo de configuración.

- pasv_min_port = 40000
- pasv_max_port = 50000

Para que solo puedan acceder usuarios registrados

- userlist_enable=YES
- userlist_file=/etc/vsftpd.userlist
- userlist_deny=NO

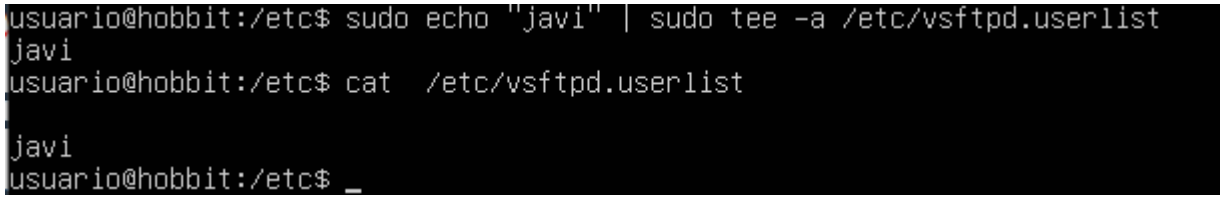
```
pasv_min_port=40000
pasv_max_port=50000
userlist_enable=YES
userlist_file=/etc/vsftpd.userlist
userlist_deny=NO_
```

VSFTPD o Very Secure FTP DAEMON

Como añadir un usuario

Para añadir un usuario, debemos de usar los comandos

- echo "nombre_usuario" | sudo tee -a /etc/vsftpd.userlist donde nombre_usuario es el nombre de usuario que queramos.
  Luego para ver el listado de los usuarios los usuarios usamos el comando.
- cat /etc/vsftpd.userlist



Para terminar reiniciaremos con sudo systemctl restart vsftpd

VSFTPD o Very Secure FTP DAEMON

Asegurar el FTP

De forma normal, FTP no hace ninguna encriptación de datos, así que se la pondremos nosotros. En primer lugar, debemos crear el certificado SSL y usarlo para proteger el servidor FTP con el comando:

- sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/vsftpd.pem -out /etc/ssl/private/vsftpd.pem

  Es bastante largo, pero necesitamos asegurar el servidor.

  -days hace que este comando sea válido para los días que le pongamos

VSFTPD o Very Secure FTP DAEMON

Ahora, abrimos el archivo de configuración.

- sudo nano /etc/vsftpd.conf

Y comentamos estas lineas

```
rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
```

Y ahora creamos en su lugar estas líneas

- rsa_cert_file=/etc/ssl/private/vsftpd.pem
- rsa_private_key_file=/etc/ssl/private/vsftpd.pem

Para apuntar al certificado que hemos creado

VSFTPD o Very Secure FTP DAEMON

Ahora agregamos las siguientes lineas:

```
allow_anon_ssl=NO
force_local_data_ssl=YES
force_local_logins_ssl=YES_
```

Para que no puedan acceder conexiones anónimas a través del SSL.

Ahora añadimos:

```
ssl_tlsv1=YES
ssl_sslv2=NO
ssl_sslv3=NO_
```

Para poder usar TLS

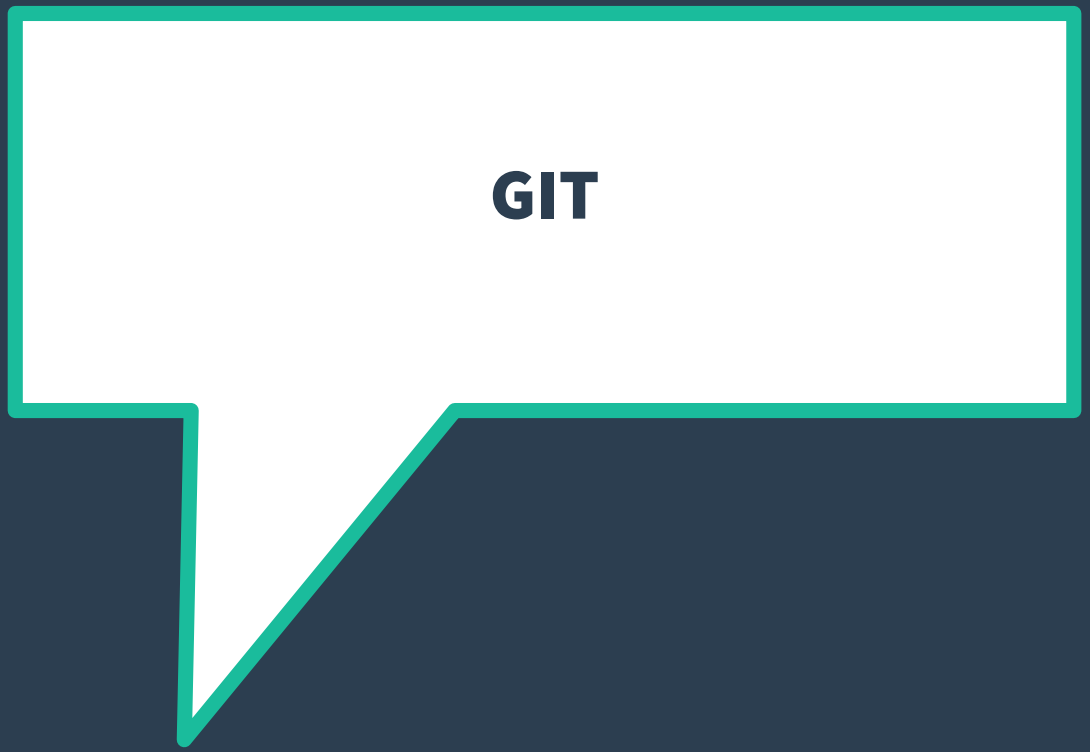
VSFTPD o Very Secure FTP DAEMON

Y por último añadiremos:

```
require_ssñ_reuse=NO
ssl_ciphers=HIGH_
```

Para que no sea necesario reutilizar SSL porque puede ocasionar que muchos clientes de FTP se averíen y la ultima instrucción utilizaremos suites de encriptación de alto cifrado, lo que significa que las longitudes de claves son iguales o superiores a 128 bits.

Para acabar reiniciaremos el sistema para guardar los cambios.


GIT

GIT

¿Que es?

Git es una herramienta que realiza una función del control de versiones de código de forma distribuida.

En Git se trabaja también con ramas. Las ramas son espacios para que el desarrollador pueda trabajar sobre un mismo proyecto sin arruinar o borrar el conjunto de archivos originales dándonos mucha flexibilidad y organización.

GIT
Como instalarlo y configurarlo.

Para instalarlo ponemos el comando

- sudo apt install git

Además de esto, tendremos que usar nuestra cuenta de git con los comandos:

- git config --global user.name "nombre"
- git config --global user.email "correo electrónico"

Si fallas, no te preocupes, puedes cambiar la configuración usando el comando nano ~/.gitconfig

GIT
Tipos de repositorios

Cuando trabajamos con Git solemos usar dos tipos de repositorios:

- Local: es el repositorio que se encuentra en tu ordenador
- Remoto: que puede estar en GitHub,GitLab...

GIT
Repositorio local

Usamos los comandos

- $ mkdir “nombre”
- $ cd “nombre”

Para crear un directorio y usamos este comando para inicializarlo:

- $ git init

```
usuario@hobbit:~$ cd test_repo
usuario@hobbit:~/test_repo$ git init
Initialized empty Git repository in /home/usuario/test_repo/.git/
```

GIT

Con esto hemos creado nuestro repositorio local, pero aun así está vacío. Para rellenarlo ponemos

- echo “tu texto”>fichero.extensión

Donde tu texto es un texto cualquiera y fichero.extension es un fichero de la extensión que quieras. Por último, para guardar las modificaciones usamos el comando:

- git add fichero.extensión

GIT
Repositorio remoto

Si queremos acceder a un repositorio remoto primero tenemos que acceder a el con el comando

- git remote add origin *http del repositorio*

Y para traerlo a nuestro ordenador usaremos:

- Git fetch origin

Ahora para inicializarlo usaremos:

- Git pull origin master

GIT

Lo que hemos hecho con estos comandos es asignarle un nombre a la http de donde cogeremos el repositorio y llamarle “origin” aunque podemos llamarle como queramos. Con fetch le decimos que el programa señale a ese repositorio y con pull lo traemos a nuestro ordenador y sincronizamos el trabajo online con el local.

Para terminar usaremos el siguiente comando para subir los cambios:

- git push

GIT
Comandos básicos

- Git clone: se usa para revisar el repertorio de repositorios locales
- Git clone usuario@ip:*ruta*: se usa para revisar el repertorio de repositorios remotos siendo usuario el nombre de usuario ip tu ip y ruta la ruta hasta el repositorio remoto.
- Git status: para ver la lista de archivos que se han cambiado y los que todavía no se han añadido o no se les ha hecho un commit
- git remote -v: para ver los repositorios remotos conectados actualmente

GIT

- git checkout -b <branch-name>: para crear una nueva rama y trabajar en ella
- git checkout <branch-name>: para pasar a trabajar en otra rama
- git branch: para ver la lista de ramas
- git branch -d <branch-name>: para borrar una rama
- git merge <branch-name>: para fusionar ramas
- git rm *nombrearchivo*: para borrar un archivo del directorio en el que estés trabajando.